UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/666,705	09/19/2003	Shelley K. Bower	200313210-1	1999

22879 7590 01/08/2007
HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2193	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	01/08/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/666,705	BOWER ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *9/19/03*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *19 September 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed 9/19/2003.

Claims 1-30 have been submitted for examination.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1, 3, 11, 16, 24, 26, 29-30 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 6, 7, 9, 15, 22, 25 of copending Application No. 10,923,192 (hereinafter '192). Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following observations in regard to the conflicting claimed subject matter.

As per **instant claim 16**, '192 claims 22 also recite parsing definitions in a description file accessible in a software manager; translating the parsed definitions into an internal program

structure for the compiled software manager; and creating a software install image based on the instantiated compiled software manager. Though '192 above claim does not recite *validation rules* belonging to the description file, '192 claim 22 recite changing the description file by changing validation rules for said file; hence the description file having validation rules therein to be changed by the user would have been a disclosed entity, or an obvious content variation of instant claim 16 rules description file.

As per **instant claims 1, 30**, '192 claims 7 and 25 also recite loading a description file from a storage accessible to a compiled program, the file not linked to the compiled program; parsing the description file with communication with configurable filter; validating the description file using the filter; and defining compiled program data based on definitions of the data structure in the description file. Although '192 claims recite a 'user interface description file' as opposed to the instant claims 'data structure description file', the connotation of description file being a data structure or user interface type of description would be analogous, or an obvious variant of one another.

As per **instant claim 3**, '192 claim 9 recites loading a description file, parsing it into a filter, validating the file; and defining structures interfaces based on the structures of description file. '192 recites applying rules to validate content loaded from said description file but does not explicitly disclose *validation rules* inside said description file; but the validation rules have been addressed as obvious, because to get appropriate objects or structure for the program via validation based on the content of the main description file, validation rules should be part of all the information loaded from this stored description file storage.

As per **instant claims 11, 24, 26, 29**, '192 claim 6, 15 recite description file with definitions, description file not linked to the compiled program; parsing and validating said file as it is read from storage; and when said file is in communication with a input filter, instantiating the program structures based on said file definitions received from said description file. Though '192 does not recite *validation rules*, claims 6, 15 recite validating the description file based on said definition being loaded, or validation status for the instantiated objects, respectively. Hence, for one skill in the art, this would render the validation rules as part of the description file (to be considered during '192 validation step) an obvious feature based on the rationale that in order to instantiate appropriate objects or structure for the program based on the content of the main description file, validation rules should be part of all the information loaded from this stored description file storage; and the product being a description file containing data and validation rules being a obvious feature of the storage of such file being loaded for validation as in '192 claims.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 1-15, 24-30 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

> The Federal Circuit has recently applied the practical application test in determining whether the claimed subject matter is statutory under 35 U.S.C. § 101. The practical application test requires that a " useful,

> concrete, and tangible result" be accomplished. An "abstract idea" when practically applied is eligible for a patent. As a consequence, an invention, which is eligible for patenting under 35 U.S.C. § 101, is in the "useful arts" when it is a machine, manufacture, process or composition of matter, which produces a concrete, tangible, and useful result. The test for practical application is thus to determine whether the claimed invention produces a "useful, concrete and tangible result".

Specifically, claim 1 recites loading a file, parsing it, validating structure inside the file and defining compiled data structures. As a whole, the claim amounts to yielding software data structures. There is not sufficient teaching as to how these abstract data structures constitute of in terms of tangible/persisted data that would be usefully applied in some real-world usage to reasonably convey that a useful, concrete and tangible application-level ultimate outcome. Thus, the claim amounts to an abstract idea and is rejected for leading to a non-statutory subject matter.

Claim 3 recites loading a structure file, parsing it, validating it into a filter and defining data structures. As set forth above, the claim amounts to steps that would not reasonably convey the realization of a concrete, tangible and useful real-world result; and is also rejected as above.

Claim 11 recites a system comprising definitions and rules description file, an input filter to parse and validate data structures and rule of said file; a compiled program configured to instantiate data structures and validation rules based on said file content. As a whole the system comprises mainly of software-implemented entities or abstract descriptive components and/or functionality without a tangible storage thereof leading to realization/execution by a real-world hardware engine upon said components/functionality to yield a tangible result as required by the Practical Application Test. The claim for being a non-practical application; and is rejected for leading to non-statutory subject matter.

Claim 24 also recites a system with description file, input filter, software manager and specification file generated from said manager. As a whole there is lack of hardware support or

tangible storage in order to reasonably convey that the software entities thus claimed can lead to execution by a tangible product/engine in order to meet the required result of the Practical Application Test. The claim is also rejected for leading to a non-statutory subject matter as set forth above for claim 11.

Claim 26 recites the same software entities of claim 24, thus is also rejected for the same reasons.

Claim 29 recites a system with description means, means for parsing and validating and a program to instantiate structures based on descriptions and rules means from the description means. What appears to be disclosed and claimed is a file and some software parser and validating means to support instantiation of data in a program; and as a system claim, the claim as a whole lacks hardware support as mentioned above. The claim is rejected for not providing sufficient teaching so as to convey that a concrete, tangible and useful result can be generated because software listed without hardware support is basically non-statutory.

Dependent claims 2, 4-10, 12-15, 25, 27-28 are also rejected for failing to remedy to the deficiency of the base claims.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

7. Claims 1-3, 5-13, 15, 24, 26-30 are rejected under 35 U.S.C. 102(a) as being anticipated by David Eng, 'Combining Static and Dynamic Data in Code Visualization', November 19, 2002, Sable Research Group, *PASTE '02*, Copyright 2002, pp. 43-50 (hereinafter Eng)

As per claim 1, Eng discloses a method for changing defined elements in a previously compiled program (*intermediate language* – ch. 1, pg. 43) using a data structure description file without modifying the compiled program, comprising the steps of:

loading a data structure description file from a storage location accessible to the compiled program, wherein the data structure description file contains definitions of data structures and is not linked into the compiled program (e.g. DTD – Fig. 1; JIL document structure - Fig. 4; *versioning of DTDs... versions of JIL* – ch. 3.3 pg. 49 – Note: version of DTD files reads on storage location for the SOOT runtime);

parsing the data structure description file for a configurable filter (e.g. ch. 2.2.1 – pg. 44 – Note: parsing definition of a DTD to yield data for a JIL reads on parsing into a filter because DTD dictates validation of proper W3c markup syntax/definition) in communication with the compiled program (Note: intermediate code as elements composing content of a JIL reads on communication with the SOOT environment – see ch. 2.2.1);

validating the data structure description file using the configurable filter (validate and identify – ch. 2.1.1; 2.2.1, pg. 44; ch. 3.1, pg. 46); and

defining the compiled program's data structures based on the definitions of the data structures in the data structure description file (Fig. 4).

As per claim 2, Eng discloses the step of validating the data description file based on language parameters loaded from the data structure description file (ch. 3.1, pg. 46).

As per claim 3, Eng discloses a method for changing data structures and validation rules in a previously compiled program using a structure and rules description file without modifying the compiled program, comprising the steps of:

loading a structure and rules description file that contains definitions of data structures and validation rules from a storage location accessible to the compiled program, wherein the structure and rules description file is not linked into the compiled program; and

parsing the structure and rules description file into a configurable filter in the compiled program; validating the structure and rules description file in the configurable filter; and

defining the compiled program's data structures and validation rules based on the definitions in the structure and rules description file;

all of which steps having been addressed in claim 1.

As per claims 5-7, Eng discloses applying validation rules within the compiled program (intermediate representation of Java class, Fig. 3) in order to check data values (Fig. 3; profiling, pg. 48); to provide enumerated values for data values (Fig. 3; enumeration, R column top - pg. 44; *collect data* - ch. 2.3.1 pg. 45); to verify that data structures follow a pattern as defined by a regular expression (Note: the checking of regular expression is inherent to any Java/markup String checking; or inherent to any W3c string syntax when validating DTD file against W3C compliance to yield a correct JIL file).

As per claims 8-9, Eng discloses validation rules in the compiled program to determine the interdependency of validation rules (Fig. 3 – Note: inheritance of hierarchy of XML reads on determining of interdependency of sub-rules among elements being parsed in view of Class

inheritance – see pg. 48) and applying interdependent validation rules based on an object's value (*profiling* – pg. 48; *count, line number,uses* - Fig. 3).

As per claim 10, Eng discloses wherein the step of validating the structure and rules description file further comprises the step of validating the structure and rules description file against a configurable filter that includes defined keywords, data structures, and a validation language (see para 2.2.1, pg. 44; XSLT bottom right , pg. 48 – Note: tag and syntax of W3C reads on keyword/structures and validation using XSLT reads on validation language).

As per claim 11, Eng discloses a system for changing data structures and validation rules in a previously compiled program using a structure and rules description file, without modifying the compiled program, comprising:

a structure and rules description file containing definitions for data structures and validation rules (refer to claim 1), wherein the structure and rules description file is not linked into the compiled program (DTD – Fig. 1; JIL document structure - Fig. 4; *versioning of DTDs... versions of JIL* – ch. 3.3 pg. 49 – Note: version of DTD files reads on storage location for the SOOT runtime);

a configurable input filter enabled to parse and validate the structure and rules description file (e.g. ch. 2.2.1 – pg. 44 – Note: parsing definition of a DTD to yield data for a JIL reads on parsing into a filter because DTD dictates validation of proper W3c markup syntax/definition) as the structure and rules description file is read from a storage location (refer to claim 1); and

wherein the compiled program is in communication with the configurable input filter, the compiled program being configured to instantiate the program's data structures and validation

rules (see Fig. 3-4; JIL document ; *Jimplex/XSL/HTML* Fig. 1) based on the definitions received from the structure and rules description file.

As per claim 12, Eng discloses a user interface (Fig. 1) in the compiled program that is configured to enable access to the data structures and validation rules.

As per claim 13, Eng discloses module to report the validation status for data structures and validation rules (Fig. 2b, d Note: results of a JIL being validated and derived from a DTD so that it comprises OO elements with status being enumerated and reported in regard the OO runtime reads on validation module) in the compiled program.

As per claim 15, Eng discloses wherein the structure and rules description file further comprises data attributes (e.g. DTD – Fig. 1; Fig. 2 – Note: DTD definitions reads on inherent attributes; and JIL derived from Class and DTD rules include OO attributes).

As per claim 24, Eng discloses a system for delivering software objects in a computing environment using data structures and validation rules that can be modified without recompilation, comprising:

a structure and rules description file containing definitions (refer to claim 1) for data structures and validation rules, wherein the structure and rules description file is not linked into the compiled software manager (DTD – Fig. 1; JIL document structure - Fig. 4; *versioning of DTDs... versions of JIL* – ch. 3.3 pg. 49 – Note: version of DTD files reads on storage location for the SOOT runtime);

a configurable input filter (e.g. ch. 2.2.1 – pg. 44 – Note: parsing definition of a DTD to yield data for a JIL reads on parsing into a filter because DTD dictates validation of proper W3c markup syntax/definition) that is configured to parse and validate the structure and rules

description file as the structure and rules description file is read from a storage location (refer to claim 1);

a compiled software manager in communication with the configurable input filter, the compiled software manager being configured to instantiate data structures and validation rules in the compiled software manager (see Fig. 3-4; JIL document ; *Jimplex/XSL/HTML* Fig. 1) based on the definitions received from the structure and rules description file; and

a product specification file generated from the compiled software manager (JIL, based on the program data structure and validation rules.

As per claim 26, Eng discloses a system for supplying data structures and validation rules in a previously compiled program using a structure and rules description file, without modifying the compiled program, comprising:

a structure and rules description file containing definitions for data structures and validation rules, wherein the structure and rules description file is not linked into the compiled program;

a parser configured to parse the structure and rules description file as the structure and rules description file is read from a storage location; and

a configurable input interpreter in the compiled program and in communication with the parser, the configurable input interpreter being configured to interpret the structure and rules description file when the compiled program is executing and configured to instantiate the program's data structures and validation rules based on the definitions received from the structure and rules description file;

all of which limitations having been addressed in claim 11 or 24.

As per claim 27, Eng discloses comprising instantiated data structures (see Fig. 2, 4) and object code to manipulate the data structures, the object code being created (retargeted ... code generator – ch. 2.1, pg. 44) by the configurable input interpreter from the definitions received from the structure and rules description file (re claim 26).

As per claim 28, refer to claim 12.

As per claim 29, Eng discloses a system for changing data structures and validation rules in a previously compiled program using a structure and rules description file, without modifying the compiled program, comprising:

a structure and rules description (refer to claim 1) means for containing definitions for data structures and validation rules, wherein the structure and rules description means is not linked into the compiled program(refer to claim 1);

a configurable input means for parsing and validating the structure and rules description means as the structure and rules description means is read from a storage location (refer to claim 1); wherein the compiled program is in communication with the configurable input means, the compiled program being configured to instantiate (refer to claims 11, 24) the program's data structures and validation rules based on the definitions received from the structure and rules description means; and

a user interface means in the compiled program that is configured to enable access to the data structures and validation rules (refer to claim 12).

As per claim 30, Eng discloses an article of manufacture, comprising:

a computer usable medium having computer readable program code means embodied therein for changing defined elements in a previously compiled program (refer to corresponding

rationale in claim 1) using a data structure description file without modifying the compiled program,

the computer readable program code means in the article of manufacture comprising: computer readable program code means for loading a data structure description file (re claim 1) from a storage location accessible to the compiled program, wherein the data structure description file contains definitions of data structures and is not linked into the compiled program (re claim 1);

computer readable program code means for parsing the data structure description file for a configurable filter in communication with the compiled program (re claim 1); computer readable program code means for validating the data structure description file using the configurable filter (re claim 1); and

computer readable program code means for defining (re claim 1) the compiled program's data structures based on the definitions of the data structures in the data structure description file.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 4, 14, and 16-23, 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over David Eng, 'Combining Static and Dynamic Data in Code Visualization'.

As per claim 4, Eng discloses information as help information stored in the structure and rules description file related to the data structure and validation rules (see *history, version, flow*

information, contributing tools – Fig. 4) but does not explicitly disclose *help files* related to the validation rules using the above information. Based on the browser-based SOOT extension of arbitrary tools for annotation or profiling support (see Ch. 2.4, pg. 45-46) by Eng, it would have been obvious for one skill in the art at the time the invention was made to provide help support inside to tool so to make use of the description file help information in Eng's browser-based visualization so that help files in the browser runtime can provide user's direction on the use of the class component during the validation in conjunction with the help information found inside the description file as mentioned above. One skill in the art would be motivated to do so because a tool (with *contributing tools* for a *version history*) that can support static analysis of data via profiling and annotation can also support help file for informing the use of help information as set forth above inside the description file such as browser help button or About button to allow retrieval of history version as taught by Eng, all of which support being well-known concepts in browser-based tools.

As per claim 14, Eng does not explicitly disclose a help system module configured to provide help information about the data structures and validation rules. But this help module or file limitation in view of Eng's framework using a browser has been addressed using the rationale as set forth in claim 4.

As per claim 16, Eng discloses a method of delivering software objects in a computing environment using a compiled software manager with data structures and validation rules that can be modified without re-compiling the software manager (e.g. *exported as JIL* -Introduction, pg. 43), comprising the steps of:

parsing data structure and validation rules read from a structure and rules description file (re claim 1) located in a storage location accessible to the software manager; translating the parsed data structure and validation rules into an internal program data structure and validation rules (e.g. ch. 2.2.1 – pg. 44 – Note: parsing definition of a DTD to yield data for a JIL reads on parsing into a internal program data because DTD dictates validation of proper W3c markup syntax/definition – see) for the compiled software manager.

Eng does not explicitly disclose creating a software install image using the compiled software manager based on the program data structure and validation rules. However, Eng disclose exporting a package to support reconstruction of code as target client (*exported as JIL* - Introduction, pg. 43; Fig. 2; IDL ch. 4.2 pg 49), hence suggests exporting of a JIL to support reconstructing of OO code for distribution in light of multi-platform portability of the JIL use via a JIMPLEX distribution (see *code generator* - ch. 4.2 pg. 49; *package of client* – ch. 4.1, pg. 49). Based on the multi-platform of XML and portability of the JIL in the JIMPLEX package for distribution, it would have been obvious for one skill in the art at the time the invention was made to provide Eng's JIMPLEX framework with capacity to build a image for installation at the client end so that such image package includes the description file enabling program structure definition and validation rules for use by the target as set forth above. One skill in the art would be motivated to do so because since JIMPLEX is a framework using portable files to support developing and validating of code at targeted client recipient of such file format, the capacity of including all necessary files inside a distribution package being part of the JIMPLEX framework would be optimizing the very capability of the same Eng's JIMPLEX framework as a enhanced product, alleviating thereby extra resources required to otherwise utilize outside products or

another middleware tools just to handle package building for the above code generation at target client.

As per claims 17-18, based on the rationale set forth in claim 16, the step of installing software components to the computing environment using the software install image created would also have obvious for the same reasons; and since Eng suggests adding necessary metadata for the development of software, such teaching (see *data extensions, metadata* -ch. 4.2, pg. 49) by Eng would be also rendering the step of *enabling additional software components to be installed into the computing environment by including additional data structure elements in the structure and rules description file* equally obvious in view of the installation package of claim 16.

As per claim 19, refer to claim 10

As per claim 20, in view of the data extensions in the JIMPLEX and portable JIL (e.g. ...any included extensions, ch. 3.1 -pg. 46) the step of adding defined keywords to the data structure as included in the structure and rules description file in view of steps 18-19 would also have been obvious.

As per claim 21, Eng discloses step of editing the structure and rules description file using a text editor to change the data structure and validation rules in the structure and rules description file (e.g. *extensions* – ch. 2.1.1, pg. 44; *annotations can be added* - ch. 2.2, pg. 44 – Note: adding extensions to DTD markup or XML document in a tool reads on having an editor).

As per claim 22, the step of including help information related to the data structure and validation rules in a separate *help file*, falls under the ambit of the rationale of claim 4; thus will incorporate the rejection as set forth therein.

As per claim 23, the providing of help access will be referred to the obvious rationale as set forth for claim 22.

As per claim 25, in regard to the system for delivering software objects as in claim 24, Eng does not teach comprising a software install image generated by the compiled software manager using the product specification file. But this limitation has been addressed in claim 16.

Conclusion

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
December 29, 2006